RAPPORT DE PROJET

Impacts EGTS à Roissy-CDG

DUBOIS Auriane
CAILLET Adrien
ROBERT Baptiste — IENAC17
BUIL Morgane — SITA

Table des matières

I/ Introduction et présentation du projet

Le système EGTS (Electric Green Taxiing System) est un système électrique implanté sur le train d'atterrissage principal d'avions, tels que l'Airbus A320 et ses dérivés, et qui permet à ceux-ci de circuler depuis leurs parkings jusqu'à la piste sans utiliser leurs moteurs principaux. Son but est de réduire la quantité de carburant utilisé par un avion, ainsi que de diminuer la quantité de gaz à effet de serre émis lors des opérations au sol.

Il présente de nombreux avantages, tel que la réduction des coûts dus à l'utilisation des réacteurs, car ceux-ci sont très consommateurs en carburant et peu efficaces au sol. Ce système étant 100% électrique, il est donc aussi respectueux de l'environnement, ce qui est un enjeu crucial dans le contexte mondial actuel. De plus ce système permettrait de ne plus utiliser de camions pour sortir les avions de leur emplacement et ainsi supprimer le temps d'attente.
L'EGTS a été conçu en 2003 par l'entreprise Delos Aerospace et est actuellement fortement développé par le groupe SAFRAN.

Cependant, ce système est très sensible aux pentes rencontrées sur le terrain aéroportuaire, contrairement aux réacteurs. Si le terrain est plat, il sera tout aussi efficace. Mais, la puissance du moteur électrique ne lui permet pas de maintenir une vitesse constante si la pente est trop importante. Ainsi, son utilisation peut fortement ralentir la circulation des avions au roulage. La réduction du trafic et les pertes d'argent qui en découleraient effraient les aéroports qui refusent donc l'utilisation de ce système.

L'objectif de notre projet est donc d'étudier l'impact du système EGTS sur l'aéroport de Roissy – Charles De Gaulle, pour ainsi déterminer si ce système est viable sur cet aéroport.
Pour se faire, nous allons tout d'abord déterminer l'altitude en tout point de l'aéroport, grâce à un algorithme de triangulation de Delaunay et au calcul des équations des plans de chacun des triangles obtenus. Puis nous implémenterons un système d'accélération qui nous permettra de déterminer les trajectoires des avions équipés de cette technologie. Enfin, nous utiliserons un algorithme de backtrack pour résoudre les conflits qui pourraient apparaître lors d'une journée « normale », si certains avions sont équipés du système EGTS.

II/ Calcul de l'altitude en tout point de l'aéroport

1/ Algorithme de triangulation de Delaunay

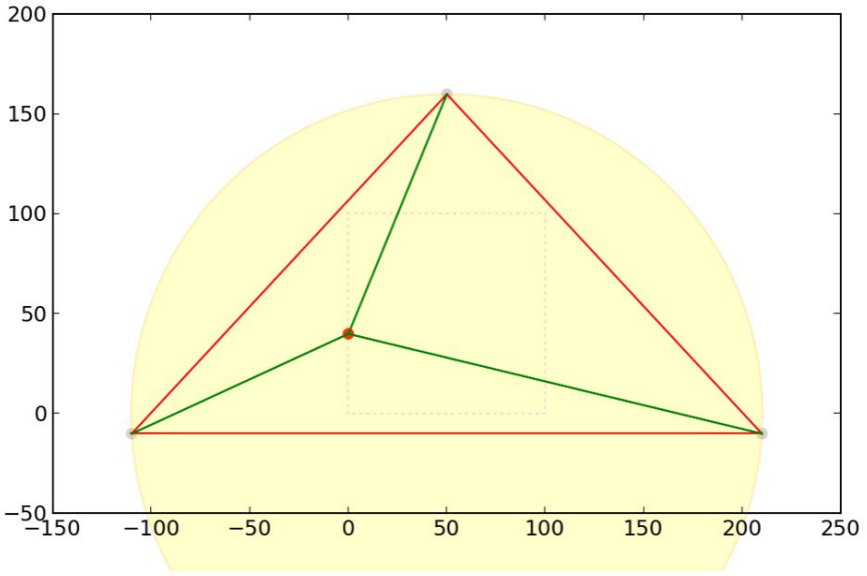
Pour calculer l'altitude de tous les points de l'aéroport, nous avons commencé par séparer le terrain en différentes « zones », grâce à un algorithme de triangulation de Delaunay.

La triangulation de Delaunay est une méthode géométrique, inventée en 1934 par le mathématicien russe Boris Delaunay, qui permet de séparer un plan en triangles, de sorte qu'aucun point du plan n'est à l'intérieur du cercle circonscrit d'un des triangles de la triangulation. Cela permet de maximiser le plus petit angle de l'ensemble des angles des triangles, évitant ainsi les triangles « allongés ».

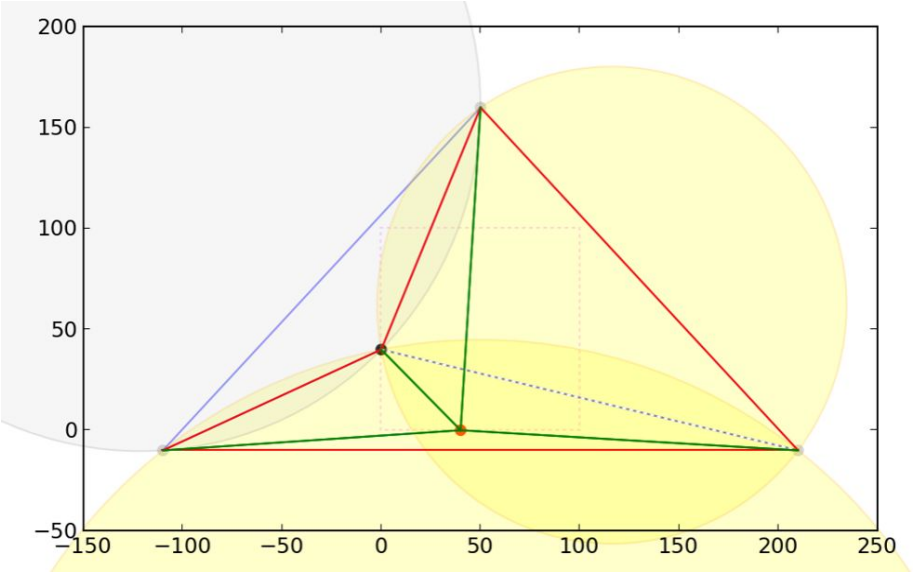
Pour effectuer cette méthode, nous avons choisi de mettre en œuvre l'algorithme de Bowyer-Watson, un algorithme incrémental qui implémente la triangulation de Delaunay. Son principe est d'ajouter des points un par un à une triangulation de Delaunay valide d'une sous-liste de points. Après chaque insertion, les triangles dont le cercle circonscrit contient le point ajouté sont supprimés, laissant alors un polygone, entourant le nouveau point, qui est alors triangulé à nouveau en utilisant ce point pour former des triangles avec les différentes arêtes du polygone.

Ci-dessous, l'explication imagée étape par étape d'une triangulation de Delaunay par l'algorithme de Bowyer-Watson :

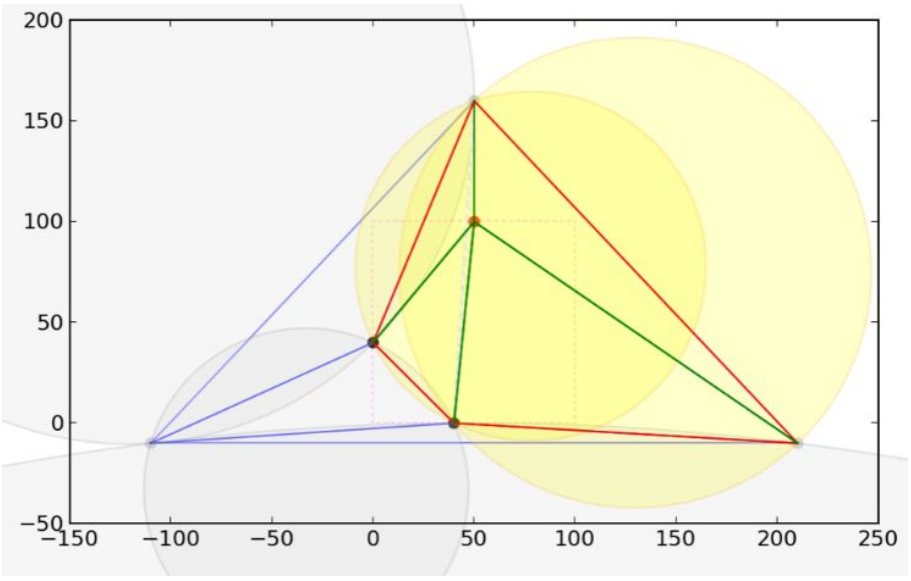
- En jaune, les cercles circonscrits des triangles posant problème
- En rouge, le polygone réalisé à partir des triangles posant problème
- En vert, les nouvelles arêtes créées (donc les nouveaux triangles)



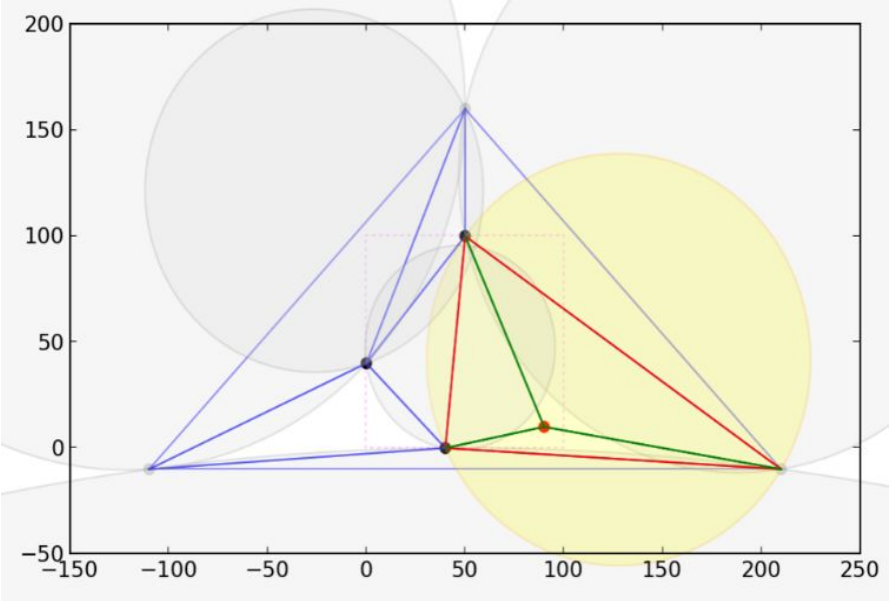
Étape 1 : création d'un super-triangle englobant et insertion du 1er point



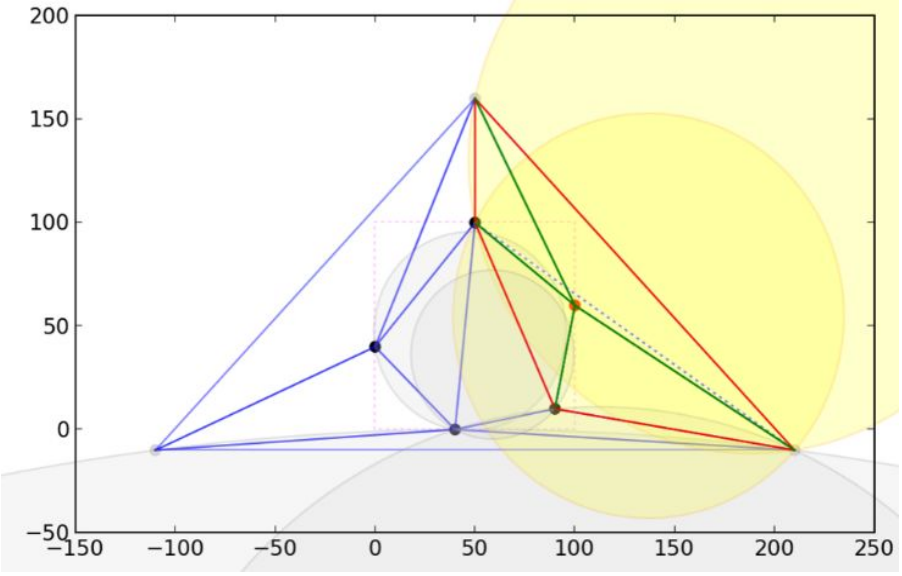
Étape 2 : insertion du second point



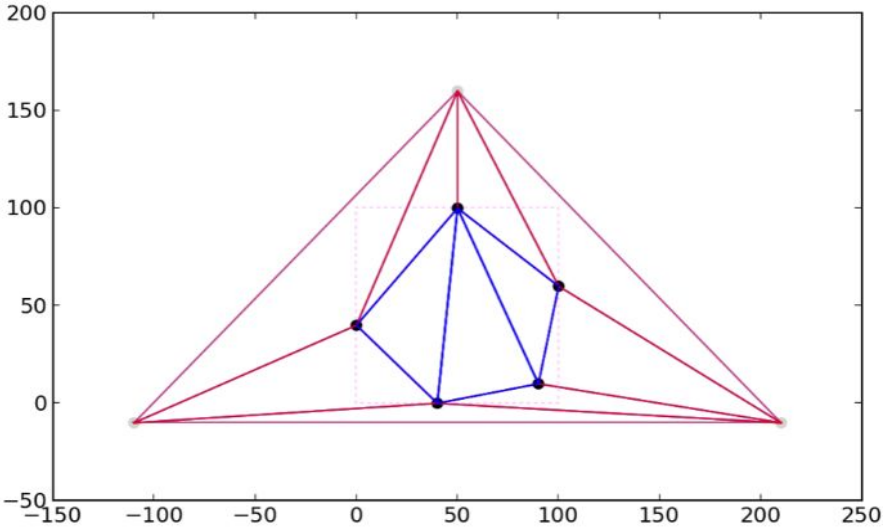
Étape 3 : insertion du troisième point



Étape 4 : insertion du quatrième point



Étape 5 : insertion du dernier point



Étape 6 : suppression des arêtes du super-triangle

On obtient alors la triangulation en bleu sur le dernier schéma.

A la fin de cet algorithme, nous conservons notre triangulation sous la forme d'une liste de triangles de telle sorte qu'un triangle soit composé de trois points (coordonnées x et y entières et z flottante) et d'un triplet de flottants (a, b, c) qui correspond à l'équation de plan $z = ax + by + c$ du triangle.

2/ Calcul du plan d'un triangle et test d'appartenance à un triangle

Maintenant que l'aéroport est divisé en triangles, nous allons pouvoir calculer l'équation de plan de chacun.
Pour ce faire, nous avons récupéré un algorithme implémentant la méthode d'élimination de Gauss. Ainsi, pour chaque triangle, à partir des coordonnées de ses trois sommets, nous avons pu déterminer les coefficients a, b et c de l'équation de plan associée : $z = ax + by + c$ en résolvant un système de 3 équations à 3 inconnues.

La méthode d'élimination de Gauss est une méthode de résolution de système de N équations à N inconnues, améliorée par le choix d'un pivot idéal à chaque itération. De ce fait, on est sûr de faire moins d'erreur sur les calculs (division par des flottants trop petits, ...) et donc que nos résultats seront le moins possible approximés.

C'est pour cela et également pour ne pas perdre inutilement de temps sur un algorithme déjà existant et largement répandu (avec de très bonnes complexités temporelle et spatiale) que nous avons choisi de récupérer une implémentation et non d'en implémenter une nous-même.

Puis nous avons créé un test, pour vérifier si un point donné appartient ou non à un triangle donné. En effet, cela sera utile par la suite pour déterminer quelle équation de plan choisir lors du calcul de l'altitude de ce point et pour pouvoir calculer des pentes entre deux points qui devront nécessairement appartenir au même triangle. De cette manière, une fois que l'on sait qu'un point appartient à un triangle, on peut calculer son altitude en se servant de l'équation du triangle.

De plus, comme un point peut appartenir à un triangle (intérieur), deux triangles (arête) ou plus (sommet), il est nécessaire d'obtenir une liste des triangles dont le point étudié fait partie. En effet, sans cette liste, nous aurions retenu seulement un des triangles possibles et peut-être que par la suite nous n'aurions pas pu calculer correctement les intersections entre notre trajectoire et les triangles de notre triangulation.

3/ Calcul des pentes sur chaque portion d'une trajectoire

Il nous a été fourni une liste des avions circulant sur l'aéroport Roissy – Charles de Gaulle lors d'une journée type. Ces avions sont notamment décrits par un ensemble de coordonnées : ces coordonnées sont celles des points (x et y) formant la trajectoire de chaque avion.

Nous avons précédemment réussi à déterminer l'équation de tous les triangles formant le secteur de l'aéroport. Il nous est donc maintenant possible de déterminer l'altitude de tous les points de l'espace à l'aide de ces équations.

Pour un point donné d'une trajectoire, s'il est prouvé comme appartenant à un triangle T (grâce au test décrit dans le point II/ 2/) d'équation $z = ax + by + c$, alors il vérifie lui-même

l’équation et il est donc aisé de déterminer sa coordonnée verticale en utilisant les coefficients a, b et c du triangle T et ses coordonnées x et y.

Lorsque les altitudes z de tous les points de toutes les trajectoires ont été déterminées, on peut alors calculer les pentes tout au long d’une trajectoire. Si deux points appartiennent à un même triangle, il suffit de diviser la différence d’altitude entre les points par la distance entre ces deux points pour obtenir un pourcentage de pente. Dans le cas où les deux points n’appartiendraient pas au même triangle, il faudra trouver le ou les points d’intersection et calculer les pentes résultantes entre chaque point. Cela est nécessaire, car le but est que la nouvelle trajectoire de l’avion colle au mieux à la modélisation 3D de l’aéroport.

Après avoir étudié les altitudes minimale et maximale de l’aéroport, on trouve :

- Altitude minimale ≈ 89m
- Altitude maximale ≈ 117.6m
- Dénivelé max ≈ 28.6m

Toutes les pentes que nous rencontrerons sur cet aéroport seront donc inférieures à ce dénivelé. Et en effet, cela correspond bien à la situation de l’aéroport Roissy - Charle De Gaulles.

III/ Implémentation du modèle d'accélération

1/ Calcul du modèle d'accélération

Notre but étant de déterminer les éventuels retards que pourraient causer le système EGTS, il nous faut donc calculer les vitesses des avions équipés d'un tel système et les comparer au système classique avec l'utilisation des réacteurs.
Nous avons donc implémenté le modèle d'accélération suivant pour déterminer la vitesse 5 secondes plus tard à partir de la vitesse courante :

- Pour le système classique, le modèle est simple : la vitesse augmente de 0.9m/s toutes les 5 secondes si l'avion n'est pas à sa vitesse maximale (au roulage). Sinon il roule à la vitesse maximale définie par la trajectoire avec un pas de temps fixe de 5s.
- Pour le système EGTS, le modèle est plus complexe : la vitesse dans 5 secondes (V_5) dépend de la vitesse courante (V), mais aussi de la masse (M) de l'avion, de la pente du terrain et bien entendu des capacités du moteur électrique.

Ce modèle permet de ne pas dépasser la vitesse et l'accélération maximales permises par le moteur électrique, alors que dans le cas d'une utilisation des réacteurs la vitesse et l'accélération auraient été plus grandes (maintien de la vitesse maximale possible malgré une plus forte pente car puissance plus grande, donc accélération atteignable plus importante).

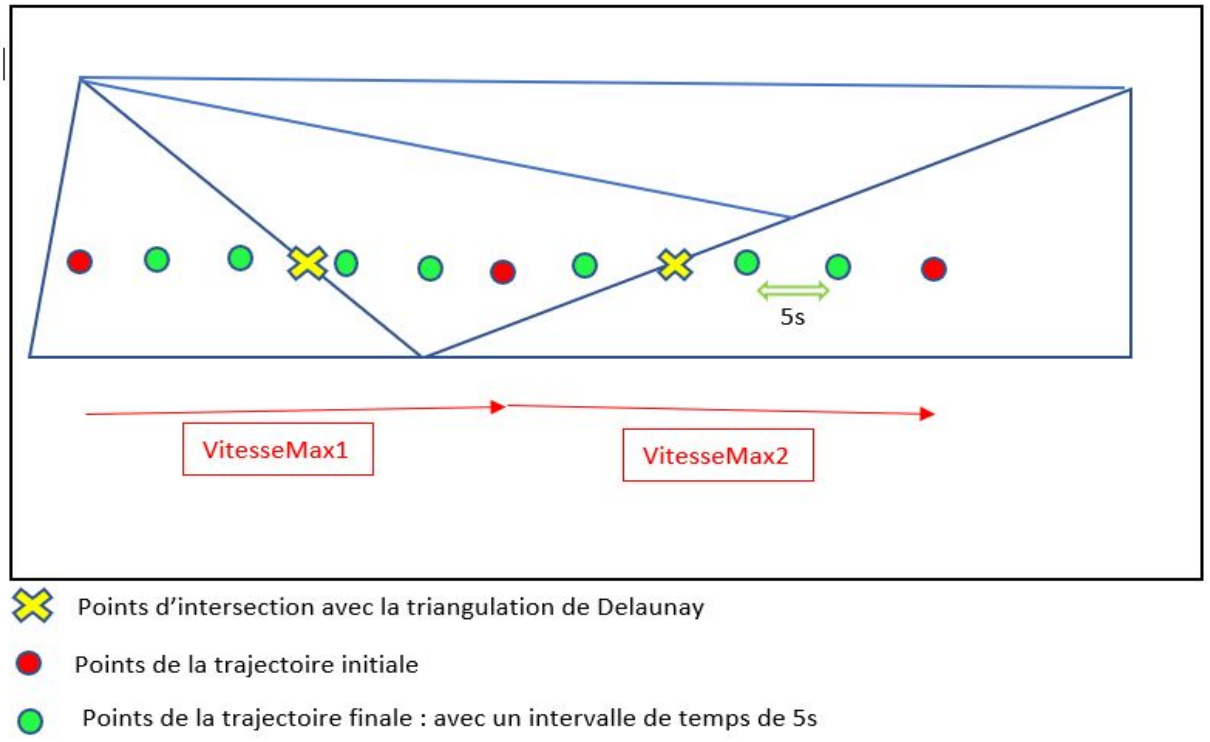
Nous pouvons alors, grâce à l'implémentation des deux modèles d'accélération, calculer les vitesses des avions sur des segments de parcours en distinguant le cas des avions avec moteurs électriques et ceux utilisant leurs réacteurs.
De plus, nous possédons pour chaque avion une vitesse maximale déterminée par sa "trajectoire initiale" constituée de points toutes les 5s. Nous savons donc que la vitesse calculée par notre modèle d'accélération ne peut pas dépasser cette vitesse maximale.

2/ Calcul des trajectoires en fonction du modèle d'accélération

Nous sommes au stade où nous possédons des points (avec leur altitude) qui forment une trajectoire et un modèle 3D de l'aéroport. Il faut maintenant adapter les trajectoires pour qu'elles suivent au mieux les différents plans de l'aéroport, définis par des triangles. Nous avons donc vérifié que deux points consécutifs appartiennent au même triangle. En effet, le but étant que la nouvelle trajectoire de l'avion colle au mieux à la modélisation 3D de l'aéroport, cela implique de suivre les différents plans de notre modélisation. Ainsi, en calculant les points d'intersection entre les différents triangles et les segments composant la trajectoire d'un avion, on peut récupérer une liste de points conformes à nos attentes. Cette liste constitue la nouvelle trajectoire, qui est sensible à la modélisation 3D de notre aéroport.

Néanmoins, même si cette idée semblait prometteuse, nous sommes confrontés à un problème de taille : comment conserver un pas de temps exploitable pour gérer la visualisation graphique mais également la gestion de conflits ?
Impossible de le faire simplement de cette manière... C'est pourquoi nous avons repris entièrement le calcul de la nouvelle trajectoire en imposant un pas de temps fixe. Désormais, en conservant la même base de recherche, nous calculons le temps que prend le parcours d'une distance entre deux points d'un même triangle. Si ce temps est supérieur au pas de temps T, on cherche alors le point situé sur le segment défini et à une distance correspondant à ce pas de temps T et on l'ajoute à la nouvelle trajectoire.

Ainsi, nous obtenons une nouvelle trajectoire constituée de points à pas de temps fixe pour les avions avec moteur électrique. Cela facilitera pour l'affichage à l'écran pour la simulation et la résolution d'éventuels conflits par la suite.



Représentation graphique du calcul d'une trajectoire telle que décrite ci-dessus

3/ Mesure et comparaison des temps de roulage

Grâce aux étapes précédentes, nous possédons pour chaque avions deux trajectoires : celle initiale avec les moteurs classiques et celle calculée avec les moteurs électriques. Nous savons que la première possède un pas de temps fixe de 5s et la seconde le pas de temps que nous avons choisi. Il est donc aisé de calculer les temps totaux que prennent chacune des trajectoires ([nombre de points – 1] * pas de temps).
Certains paramètres sont également à prendre en compte : un avion avec des moteurs normaux perdra du temps lors de la sortie de sa place de parking tandis que celui possédant un moteur électrique pourra partir directement (pas de camion pour l'aider à sortir de son parking (pushback)).
Grâce à l'interface, on peut également visualiser les deux trajectoires et voir laquelle est la plus rapide.

En lançant la simulation, on remarque que les avions avec des moteurs électriques vont bien moins vite que les avions utilisant leurs réacteurs.
Pour un certain avion donné et une trajectoire donnée, on peut voir, par exemple, que lorsqu'il est équipé de réacteurs, l'avion met 665s (11min 5s) et lorsqu'il utilise un moteur électrique, il met 720s (12 min).
En moyenne, nous trouvons donc que les avions avec moteurs électriques mettent 10% de temps supplémentaire à parcourir les mêmes trajectoires.

IV/ Résolution des conflits

1/ Algorithme de backtrack

Maintenant que toutes les trajectoires ont été obtenues, il est intéressant, pour parfaire notre modèle et affiner notre réponse à la question de l'influence des moteurs électriques sur les aéroports, de se poser les questions suivantes : comment gérer les conflits créés par le fait que les avions équipés de moteurs électriques vont moins vite que ceux utilisant leurs réacteurs ? Ou encore, quelle sont les conséquences du fait que les avions avec moteurs électriques n'attendent pas quelques minutes entre la sortie de leur place de parking et leur départ pour rejoindre la piste ?

Nous avons donc implémenté un algorithme de backtrack pour résoudre les situations de conflit en imposant la règle suivante : en cas de conflit entre deux avions, l'avion qui est prioritaire au départ (heure de décollage inférieure) continue sa route tandis que l'autre s'arrête un temps t puis reprend son chemin lorsque le conflit est résolu.

Pour faciliter le calcul des trajectoires, nous avons choisi, en cas de conflit, de faire attendre un des deux avions 5s (temps entre chaque point de notre simulation). Si au bout des 5s le conflit n'est pas résolu, l'avion attendra 5s supplémentaires et ainsi de suite.

Par manque de temps pour optimiser la fonction et à cause de difficultés rencontrées en voulant calculer les trajectoires point à point, il a été également choisi de travailler avec les trajectoires totales. En effet après calcul de la trajectoire totale d'un avion, on regarde si à chaque pas de temps il est en conflit avec les autres avions (ceux déjà résolus, i.e. dont les trajectoires sont sûres et ne changeront plus).

- Si l'avion n'est pas en conflit à ce temps t : le point est ajouté à sa trajectoire définitive et on passe au point suivant (pas de temps suivant).
- S'il est en conflit à ce temps t : on répète le point précédent (pour simuler une attente de 5s) et on l'ajoute à la trajectoire définitive. On réinitialise alors la vitesse à 0 (arrêt de l'avion) puis on recalcule toute la trajectoire possible de l'avion (puisque la vitesse a changée).

Néanmoins, cette fonction étant beaucoup trop longue, nous avons choisi un compromis en ne considérant qu'une liste de points calculés entre deux points de la trajectoire initiale. Cela permet de réduire les calculs de trajectoires en cas de conflits (on traite une liste de 4-5 points au lieu d'une liste de 100 points environ). Mais nous avons eu des problèmes avec des compteurs de temps permettant de savoir où nous nous situions dans notre trajectoire. Nous avons alors décidé de reprendre notre première idée.

Également par manque de temps, la question de l'occupation des pistes n'a pas été prise en compte. Nous nous sommes en effet concentré sur les mouvements des avions au roulage, sans tenir compte des contraintes de la piste (décollage et atterrissage).

2/ Mesure des temps de roulage et des retards

Après avoir calculé les trajectoires pour permettre d'éviter les conflits, nous pouvons réévaluer les nouveaux temps totaux et les comparer à nouveau. En comparant ces nouveaux résultats aux anciens, nous pouvons alors en déduire le retard dû à la résolution de conflits.

Précédemment, les temps obtenus ne tenaient pas compte des autres avions au roulage. En effet, nous avions juste comparé pour un même avion le temps mis pour parcourir sa trajectoire avec des réacteurs puis avec des moteurs électriques. Désormais, nous allons tenir compte de l'impact d'un avion avec moteurs électriques sur l'ensemble des autres avions au roulage.

Il semblerait que les avions avec moteurs électriques ralentissent le trafic, notamment lors des heures de pointe. Néanmoins, notre algorithme de backtrack n'étant pas totalement au point, nous ne pouvons pas confirmer avec certitude ce résultat. Mais on peut déjà constater, en visionnant la simulation sans backtrack, que beaucoup d'avions se "doublent", surtout des avions avec réacteurs suivant des avions électriques. On peut donc penser qu'en imposant à ces avions de s'arrêter, nous allons ralentir le trafic au roulage.

V/ Simulation

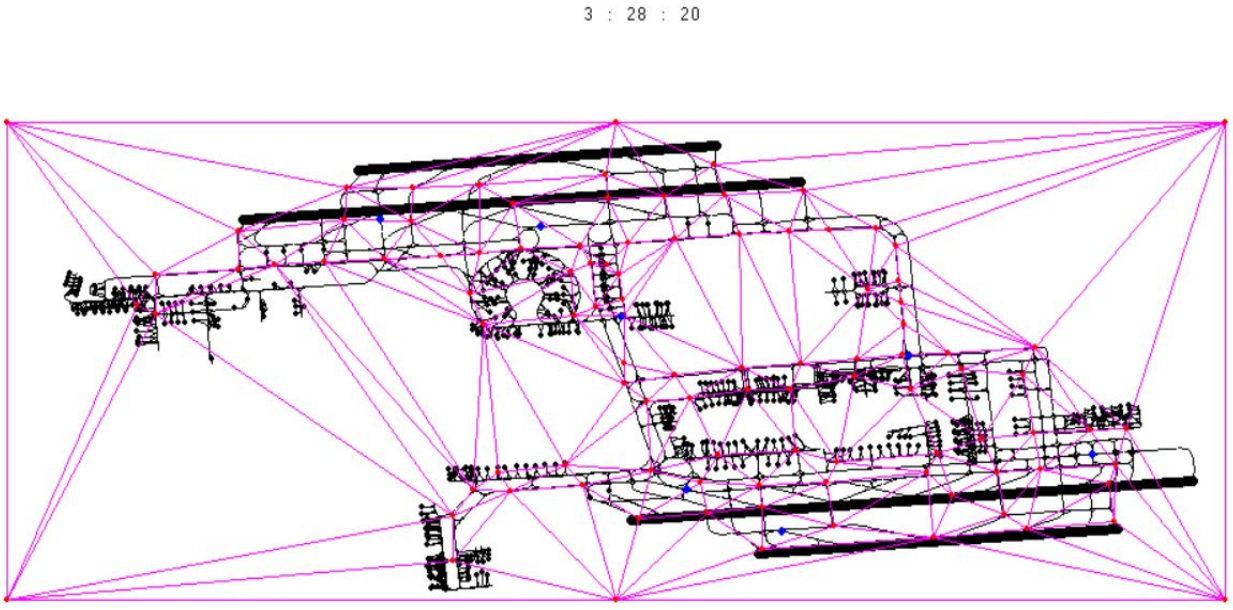
En nous inspirant de l'application Pyairport vue en TP l'année dernière, nous avons recréé une visualisation de l'aéroport. A cette image, nous avons ajouté la visualisation de notre triangulation de Delaunay. Puis nous avons construit la simulation, de manière à afficher à l'écran les mouvements des avions selon leurs heures de départ et d'arrivée.

Ainsi, après le calcul des nouvelles trajectoires et la résolution des conflits, cet affichage graphique permet de vérifier que les trajectoires sont bonnes (pas de sortie de runway), que les vitesses sont cohérentes et que les conflits sont bien résolus (pas d'avions se chevauchant ou rattrapant celui d'avant).

Sur cette simulation, nous avons choisi de laisser la triangulation de Delaunay apparente (en rose) et de différentier les avions électriques des non-électriques en leurs attribuant des couleurs :

- Bleu pour les avions "normaux"
- Vert pour les avions électriques

Nous avons utilisé la librairie Graphics pour créer cette simulation. Néanmoins, Graphics n'est pas optimisé pour afficher de nombreux objets à l'écran et les faire bouger. La fenêtre graphique est composée d'une mémoire et d'un affichage : c'est en dessinant seulement sur l'affichage puis en réactualisant l'affichage à l'aide de la mémoire que l'on arrive à créer une animation.



Affichage de notre simulation

VI/ Conclusion

Notre projet était d'étudier le système EGTS sur l'aéroport de Roissy – Charles De Gaulle et de déterminer si ce système est viable sur cet aéroport.

Nous avons montré que les avions avec moteurs électriques mettaient en moyenne 10% de temps en plus que les avions utilisant leurs réacteurs. Et ce, sur des pentes à faibles dénivelés (28.6m au maximum). Il est également clair que plus les pentes seront fortes, plus la différence de temps de parcours entre les avions à moteurs électriques et les avions utilisant leurs réacteurs sera forte. Le gain de temps lors du pushback (qui n'a plus lieu d'être avec un moteur électrique) permet de limiter cette perte de temps mais n'est pas suffisant. En effet, dès que l'on met en place une circulation plus importante sur l'aéroport, les avions électriques ralentissent encore plus le trafic. Les avions allant moins vite, ils ralentissent les autres, surtout en heure de pointe, et créent de grands décalages de temps en fin de journée.

On peut donc en conclure que même si les avions avec moteurs électriques n'ont pas un temps de roulage beaucoup plus élevé que les autres (en tout cas sur un dénivelé relativement faible), lors des heures de pointes ils ralentissent quand même considérablement le trafic au roulage.

Néanmoins, notre simulation ne couvre pas tous les aspects.
Tout d'abord, notre modèle de décélération n'est pas parfait : nous avons choisi de repasser la vitesse à zéro et d'attendre en cas de conflit, mais en réalité l'avion devrait suivre un modèle de décélération semblable à celui d'accélération. De même, les avions pourraient juste ralentir et ne pas totalement s'arrêter en cas de conflit (cela serait plus proche de la réalité).
Ensuite, nos avions suivent toujours la même trajectoire (les mêmes runway, ...) mais en cas de conflit, une solution de résolution pourrait être de changer les trajectoires et d'exploiter différemment l'aéroport (donner d'autres emplacements de stationnement, ...).
En outre, notre modèle d'accélération (et notre simulation) ne tient pas compte de certains aspects pouvant également impacter le déplacement des avions (météo, ...) et la simulation ne considère qu'un type d'avions très courant (famille A320). Il faudrait également étudier le déplacements d'autres types d'avions possiblement présents sur cet aéroport international.
Enfin, notre code en lui même pourrait être amélioré en résolvant des problèmes de complexité n'ayant pu être pris en compte par manque de temps (parcours de listes alors que l'on pourrait peut-être imaginer une manière de stocker une information supplémentaire permettant d'éviter de tout parcourir, ...). Etablir un graphe des triangles pourrait être une idée pour ne pas avoir à chercher le bon triangle parmis toute la liste de la triangulation, mais seulement parmis 3 ou 4 triangles voisins, lors du calcul de la trajectoire.

En conclusion, l'ajout de moteurs électriques aux avions au roulage semble viable à première vue, même s'ils ralentissent le trafic aux heures de pointes. Cela pourrait être validé ou rejeté par une étude plus complète avec une exploitation différente de l'aéroport pour mieux les intégrer (en prenant en compte qu'ils sont moins rapides).

VII/ Bibliographie

Sites web :

1. Wikipedia, the free encyclopedia. EGTS. Dernier accès le 12 décembre 2018 à l'adresse *https://en.wikipedia.org/wiki/EGTS*
2. Wikipedia. Triangulation de Delaunay. Dernier accès le 12 décembre 2018 à l'adresse *https://fr.wikipedia.org/wiki/Triangulation_de_Delaunay*
3. Wikipedia, the free encyclopedia. Bowyer-Watson algorithm. Dernier accès le 12 décembre 2018 à l'adresse *https://en.wikipedia.org/wiki/Bowyer%E2%80%93Watson_algorithm*
4. Rosetta Code. Gaussian elimination. Dernier accès le 15 décembre 2018 à l'adresse *https://rosettacode.org/wiki/Gaussian_elimination*

5. Notre projet. *https://github.com/22totoche22/BE_Impacts_EGTS_a_Roissy-CDG*